AMENDMENTS TO THE CLAIMS

Please cancel claims 4 and 34 without prejudice. Please accept amended claims 1, 3, 5-7, 9, 19, 28, 31, 33, 35-37 and 39, and new claims 40 and 41, as follows:

1. (Currently Amended) A method for renaming memory references to stack locations in a computer processing system comprising a plurality of processors, comprising the steps of:

fetching, by a first processor, an instruction referencing a stack;

~~detecting stack references that use architecturally defined stack access methods~~;

replacing the ~~stack references~~ instruction referencing the stack with a references to a processor-internal registers of the first processor and entering the reference to the processor-internal register in a dispatch table upon determining that the instruction uses an architecturally defined stack access method; and

~~synchronizing an architected state between the processor-internal registers and a main memory of the computer processing system~~ performing a consistency-preserving operation to recover an in-order value for the instruction referencing the stack from a main memory, and entering the in-order value in the dispatch table upon determining that the stack reference references a register of a second processor.

2. (Cancelled)

3. (Currently Amended) The program storage device according to claim ~~1~~ 40, wherein said synchronizing step comprises the step of inserting in-order write operations for all of the stack references that are write stack references.

4. (Cancelled)

5. (Currently Amended) The method according to claim [[4]] 1, wherein said step of performing ~~a~~ the consistency-preserving operation comprises the step of bypassing a value from a given processor-internal register to a load operation that references a stack area and that does not use the architecturally defined stack access methods.

6. (Currently Amended) The method according to claim [[4]] 1, further comprising the step of synchronizing an architected state between the processor-internal registers and a main memory of the computer processing system, and wherein said step of performing ~~a~~ the consistency-preserving operation ~~comprises the step of recovering an~~ recovers the in-order value for the stack reference from the main memory, upon performing said synchronizing step.

7. (Currently Amended) The method according to claim 6, wherein the in-order value is written to the main memory by an in-order write operation inserted into an instruction stream containing an instruction corresponding to the ~~stack reference~~ instruction referencing the stack, when the ~~stack reference~~ instruction referencing the stack is a write stack reference.

8. (Original) The method according to claim 6, further comprising the step of writing the in-order value to the main memory in response to a load operation that does not use the architecturally defined stack access methods.

9. (Currently Amended) The method according to claim 1, wherein said step of performing a consistency-preserving operation comprises the steps of:

discarding all out-of-order state;

synchronizing an architected state between the processor-internal registers and ~~a~~ the main memory of the computer processing system; and

restarting execution after a store operation has been performed that does not use the architecturally defined stack access methods.

10. (Original) The method according to claim 1, wherein the architecturally defined stack access methods comprise memory accesses that use at least one of a stack pointer, a frame pointer, and an argument pointer.

11. (Original) The method according to claim 1, wherein the architecturally defined stack access methods comprise push, pop, and other stack manipulation operations.

12. (Previously Presented) A method for renaming memory reference to stack locations in a multiprocessor computer processing system, comprising the steps of:

determining whether a load instruction references a location in a local stack using an architecturally defined register for accessing a stack location, wherein each processor comprises a respective local stack;

determining whether a rename register exists for the references location in the local stack, when the load instruction references the location using the architecturally defined register; and

replacing the reference to the location by a reference to the rename register, when the rename register exists.

13. (Original) The method according to claim 12, wherein the architecturally defined register corresponds to a pointer for accessing stack locations.

14. (Original) The method according to claim 13, wherein the pointer for accessing the stack locations is one of a stack pointer, a frame pointer, and an argument pointer.

15. (Original) The method according to claim 12, wherein the architecturally defined stack access methods comprise push, pop, and other stack manipulation operations.

16. (Original) The method according to claim 12, wherein said step of determining whether the renaming register exists comprises the step of computing one of a symbolic address and an actual address of the location.

17. (Original) The method according to claim 12, wherein said step of determining whether the rename register exists is performed during one of a decode, and address generation, and a memory access phase.

18. (Previously Presented) The method according to claim 12, further comprising the steps of:

determining that the rename register does not exist; and

performing the load instruction from one of a main memory and a cache of the system.

19. (Currently Amended) The method according to claim 12, further comprising the step of determining, by a first processor, whether the load instruction references a location in any stack, including the local stack, using ~~another~~ a register of a second processor, when the load instruction does not reference the location using the architecturally defined register.

20. (Original) The method according to claim 19, wherein said step of determining whether the load instruction references the location using the other register comprises the step of marking translation lookaside buffer (TLB) entries of pages in the local stack as containing stack references, when the load instruction references the location using the other register.

21. (Original) The method according to claim 19, further comprising the step of performing the load instruction from one of a main memory and a cache of the system, when the load instruction does not reference the location using the other register.

22. (Original) The method according to claim 19, further comprising the step of executing a consistency-preserving mechanism to perform the load instruction from the stack area, when the load instruction references the location using the other register.

23. (Original) The method according to claim 12, further comprising the step of allocating a rename register for the location, when the rename register does not exist.

24. (Original) The method according to claim 23, further comprising the step of inserting an operation, into an instruction stream containing the load instruction, to load the location from a processor of the system to the allocated rename register, upon allocating the rename register.

25. (Original) The method according to claim 24, further comprising the step of replacing the reference to the location by a reference to the allocated rename register, upon inserting the operation.

26. (Previously Presented) A method for renaming memory reference to stack locations in a multiprocessor computer processing system, comprising the steps of:

determining whether a store instruction references a location in a local stack using an architecturally defined register for accessing a stack location, wherein each processor comprises a respective local stack;

allocating a rename register for the location, when the store instruction references the location using the architecturally defined register; and

replacing the reference to the location by a reference to the rename register.

27. (Original) The method according to claim 26, further comprising the step of inserting an operation, into an instruction stream containing the store instruction, to store the location from the rename register to a main memory of the system, upon replacing the reference to the location by the reference to the rename register.

28. (Currently Amended) The method according to claim 26, further comprising the step of determining, by a first processor, whether the store instruction references a location in any stack, including the local stack, using ~~another~~ a register of a second processor, when the store instruction does not reference the location using the architecturally defined register.

29. (Original) The method according to claim 28, further comprising the step of performing the store instruction from one of a main memory and a cache of the system, when the store instruction does not reference the location using the other register.

30. (Original) The method according to claim 28, further comprising the step of executing a consistency-preserving mechanism to perform the store instruction to the stack area, when the store instruction references the location using the other register.

31. (Currently Amended) A program storage device readable by machine, tangibly embodying a program of instructions executable by the machine to perform methods steps for renaming memory references to stack locations in a computer processing system comprising a plurality of processors, the method steps comprising:

fetching, by a first processor, an instruction referencing a stack;

~~detecting stack references that use architecturally defined stack access methods~~;

replacing the ~~stack references~~ instruction referencing the stack with a references to a processor-internal registers of the first processor and entering the reference to the processor-internal register in a dispatch table upon determining that the instruction uses an architecturally defined stack access method; and

~~synchronizing an architected state between the processor-internal registers and a main memory of the computer processing system~~ performing a consistency-preserving operation to recover an in-order value for the instruction referencing the stack from a main memory, and entering the in-order value in the dispatch table upon determining that the stack reference references a register of a second processor.

32. (Cancelled)

33. (Currently Amended) The program storage device according to claim ~~31~~ 41, wherein said synchronizing step comprises the step of inserting in-order write operations for all of the stack references that are write stack references.

34. (Cancelled)

35. (Currently Amended) The program storage device according to claim ~~34~~ 31, wherein said step of performing a consistency-preserving operation comprises the step of bypassing a value from a given processor-internal register to a load operation that references a stack area and that does not use the architecturally defined stack access methods.

36. (Currently Amended) The program storage device according to claim ~~34~~ 31, further comprising the step of synchronizing an architected state between the processor-internal registers and a main memory of the computer processing system, and wherein said step of performing ~~a~~ the consistency-

preserving operation ~~comprises the step of recovering an~~ recovers the in-order value for the stack reference from the main memory, upon performing said synchronizing step.

37. (Currently Amended) The program storage device according to claim 36, wherein the in-order value is written to the main memory by an in-order write operation inserted into an instruction stream containing an instruction corresponding to the ~~stack reference~~ instruction referencing the stack, when the ~~stack reference~~ instruction referencing the stack is a write stack reference.

38. (Original) The program storage device according to claim 36, further comprising the step of writing the in-order value to the main memory in response to a load operation that does not use the architecturally defined stack access methods.

39. (Currently Amended) The program storage device according to claim ~~34~~ 31, wherein said step of performing a consistency-preserving operation comprises the steps of:

discarding all out-of-order state;

synchronizing an architected state between the processor-internal registers and ~~a~~ the main memory of the computer processing system; and

restarting execution after a store operation has been performed that does not use the architecturally defined stack access methods.

40. (New) The method according to claim 1, further comprising synchronizing an architected state between the processor-internal registers and a main memory of the computer processing system.

41. (New) The program storage device according to claim 31, further comprising synchronizing an architected state between the processor-internal registers and a main memory of the computer processing system.